Kunskapskontroll 2, Deep Learning.

Del 1: teoretiska frågor.

1. **Hur är AI, Maskininlärning och Deep Learning relaterat?**
   AI eller Artificiell Intelligens är det övergripande området som syftat på att få datorer eller system att efterleva mänsklig intelligens. Göra sådant som normalt mänsklig intelligens behövs till, som tex förstå språk eller känna igen bilder.
   Ett delområde inom AI är Maskininlärning, som bygger på idén att datorer kan lära sig själv genom att se mönster från data. Utan att uttryckligen programmeras för varje uppgift den ska lösa. Modellen tränas på att hitta mönster utifrån data och förbättrar sin prestation över tid, modellen anpassas med träning för att kunna återge mönstren från tränings data på ny osedd data.
   Deep Learning är i sin tur en specialiserad gren inom maskinlärning. Som använder artificiella neurala nätverk med många lager för att lära sig komplexa mönster, utifrån data. Vilket är inspirerat av hur mänskliga hjärnan är uppbyggd. Detta kan användas för att tex känna igen ansikten eller översätta språk.

2. **Hur är Tensorflow och Keras relaterat?**
   Tensorflow är ett omfattande ramverk (bibliotek) för maskininlärning och deep learning, utvecklat av Google. Keras är ett högnivå-API (ett enkelt gränssnitt) för att bygga och träna neurala nätverk.
   Keras är integrerat i Tensorflow, när du använder Keras så är det Tensorflow som jobbar i bakgrunden, som sköter beräkningar, optimering och GPU stöd. Medans Keras bygger modellen, arkitektur, lager och träningslogik.
   Keras är ett användarvänligt gränssnitt för att arbeta med neurala nätverk, medan Tensorflow är motorn som kör dem.

3. **Vad är en parameter? Vad är en Hyperparameter?**
   Parameter är de värden som modellen själv justerar och lär sig under träningen. I ett neuralt nätverk är detta vikter mellan noder samt bias (förskjutningen). Dessa parametrar uppdateras automatiskt genom optimering, (tex med gradient descent) för att förbättra modellens noggrannhet.
   Exempel: I en dense-layer med 10 neuroner och 5 inputs, har du 10*5 = 50 vikter, + 10 bias, vilket ger totalt 60 parametrar.

   Hyperparametrar är inställningar som du själv bestämmer innan träningen. De styr hur modellen lär sig, men de uppdateras inte automatiskt under träningen.
   Exempel: Antal lager – hur djup modellen är, antal Neuroner – hur bred varje lager är, inlärningshastighet – hur snabbt modellen justerar sina vikter. Batch storlek –

hur många exempel tränas samtidigt. Epochs – hur många gånger hela datan tränas igenom, Dropoute rate – hur mycket som slumpslås av vid träning.

4. **När man skall göra modellval och modellutvärdering kan man använda tränings-, validerings- och testdataset. Förklara hur de olika delarna kan användas.**
   Om du har ett dataset på tex tusen observationer. Som ska användas för att träna en modell, så är ett vanligt tillvägagångsätt att dela upp datasätet i tre delar. Ofta 60% träningsdel, 20% valideringsdel och 20% testdel.
   En träningsdel som modellen tränas på, där lär sig modellen att själv hitta mönster och justera sina parametrar (bias och vikter mellan noder) utifrån träningsdatan.
   Valideringsdelen används till att utvärdera modellen efter att den tränats på träningsdatan. Baserat på resultaten från valideringsdatan, så justerar man hyperparametrarna (antal lager, antal neuroner osv) för att förbättra modellen. Används även för att välja vilken modell man ska använda, gå vidare med.
   Testdelen används för att utvärdera hur modellen presterar på helt osedd data, när modellen är helt färdigtränad. Ger en slutgiltig och objektiv bedömning av modellens prestanda. Testdatan ska aldrig påverka träningen eller modellvalet.

5. **Förklara vad koden gör:**
   n_cols = x_train.shape[1]
   *# Räknar ut **antal input-funktioner (features)** i datan – behövs för att veta input-dimensionen.*

   nn_model = Sequential()
   *# Skapar en **sekventiell modell**, vilket betyder att lagren läggs på varandra i ordning.*

   nn_model.add(Dense(100, activation='relu', input_shape=(n_cols,)))
   *# Lägger till ett **Dense-lager** (fullt kopplat):*

- *100 neuroner*
- *ReLU som aktiveringsfunktion*
- *Tar emot n_cols inputs*

   nn_model.add(Dropout(rate=0.2))
   *# Slumpar bort **20 % av neuronerna** under varje träningssteg → skyddar mot **överträning***

   nn_model.add(Dense(50, activation='relu'))
   # Ett andra Dense-lager med 50 neuroner och ReLU

```
nn_model.add(Dense(1, activation='sigmoid'))
```
Utgångslager*:*

- *1 neuron (eftersom det är binär klassificering)*
- *sigmoid gör att output blir mellan 0 och 1 → tolkas som sannolikhet för klass 1*

```
nn_model.compile(
optimizer='adam',
loss='binary_crossentropy',
metrics=['accuracy' ])
```
 # adam = en smart optimeringsalgoritm som justerar vikter automatiskt
binary_crossentropy = används för binära klassificeringsproblem
accuracy = loggar hur ofta modellen gissar rätt

```
early_stopping_monitor = EarlyStopping(patience=5)
```
 # Stoppar träningen automatiskt om valideringsprestandan inte förbättras på 5 epoker i rad

```
nn_model.fit(
 x_train,
 y_train,
 validation_split=0.2,
 epochs=100,
 callbacks=[early_stopping_monitor])
```
Tränar modellen på x_train, y_train
20 % av datan används som valideringsdata
Max 100 epoker, men kan avbrytas tidigare om modellen slutar förbättras
EarlyStopping kontrollerar överträning

Sammanfattning:
Den här koden:

1. Skapar ett neuralt nätverk med:
   2 gömda lager (100 + 50 neuroner)
   Dropout
   Sigmoid-output
2. Tränar modellen för binär klassificering
3. Stoppar träningen tidigt om modellen inte förbättras på valideringsdatan

6. **Vad är syftet med att regularisera en modell?**
   Syftet med regularisering är att minska risken för överträning. Att modellen blir för

komplex eller att den lär sig brus eller detaljer som är för anpassade enbart för träningsdatan och inte passar in på ny osedd data. Vilket kan ske om modellen tränas länge på träningsdata utan regularisering. Man vill att modellen fokuserar på de grova viktigaste mönstren i datan.
Exempel på typer av regularisering som kan användas är:
L1/L2(Ridge/ Lasso)- lägger till en straffterm i förlustfunktionen baserat på vikternas storlek.
Dropout – Slumpar bort vissa neuroner under träning, tvingar nätverket att bli robust.
Early stopping – Stoppar träningen tidigt om valideringsresultatet slutar förbättras.
Data augummentation – Skapar variation i träningsdata för att motverka överinlärning.

7. **”Dropout” är en regulariseringsteknik, vad är det för något?**
   Vid varje träningssteg (epoch) väljs ett slumpmässigt antal neuroner i nätverket bort, de släcks ned så att de inte används i just det träningspasset. Detta leder till att nätverket inte kan lita för mycket på enskilda neuroner, utan tvingas lära sig robusta mer generella mönster, vilket förhindrar att modellen överanpassar sig till träningsdatan.
   I slutsteget i själva testningen så används alla neuroner, dropout sker endast under träningen.

8. **”Early stopping” är en regulariseringsteknik, vad är det för något?**
   Stoppar träningen av en modell innan den börjar överträna. Modellen utvärderas efter varje epok på valideringsdata, om valideringsförlusten (val_loss) slutar förbättras, tolkas det som att modellen inte längre lär sig att generalisera. Då avbryts träningen.

9. **Din kollega frågar dig vilken typ av neuralt nätverk som är populärt för bildanalys, vad svarar du?**
   CNN (Convolutional Neural Networks) är en specialiserad typ av neuralt nätverk som är särskilt bra på att analysera visuella mönster i bilder genom att identifiera kanter, färger, former och objekt. Upptäcka lokala mönster tex ögon, mun, hjul osv.
   CNN använder färre parametrar än vanliga dense-nätverk vilket effektiviserar träningen.

10. **Förklara översiktligt hur ett ”Convolutional Neural Network” fungerar.**
    CNN består av flera lager som extraherar och tolkar mönster i en bild steg för steg, hierarkiskt från enkla till mer komplexa.

Input layer, tar in en bild som en matris av pixlar tex 28x28 bild.
Convolutional layer använder små filter/kärnor som rör sig över bilden, identiferar lokala mönster, tex hörn, kanter färgområden osv.
Pooling lager reducerar stoleken på feature maps, behåller viktiga mönster och slänger onödig information.
Feature maps omvandlas till en lån vektor för att kunna skickan in i Dense lagret, som använder den extraherade informationen för att fatta beslut. Vanligtvis det sista lagret ger klassificerings med softmax eller sigmoid. Allt sker automatisk genom träning på många bilder.
CNN fungerar genom att:
Analysera lokala mönster i bilden med konvolutioner
Minska datakomplexitet med pooling.
Fatta beslut i dense-lager

11. **Vad gör nedanstående kod:**

```
model.save("model_file.keras")
# Sparar den tränade modellen till en fil som heter model_file.keras.
Innehåller:
Modellens arkitektur (lager, aktiveringsfunktioner m.m.)
Vikterna (det modellen har lärt sig)
Optimerare och eventuella inställningar

my_model = load_model("model_file.keras")
# Laddar tillbaka modellen från filen och lagrar den i variabeln my_model.
Du kan nu använda modellen direkt för:
        Prediktion (my_model.predict(...))
        Fortsatt träning (my_model.fit(...))
        Utvärdering (my_model.evaluate(...))
```

Detta används när man vill. Spara modellen efter träning och använda den senare. Undvika att träna om modellen varje gång eller dela modellen med andra. Använda modellen i en webbapplikation (t.ex. i en Streamlit-app).

12. **Deep Learning modeller kan ta lång tid att träna, då kan GPU via t.ex. Google Colab skynda på träningen avsevärt. Skriv mycket kortfattat vad CPU och GPU är.**
    CPU (Central Processing Unit) är datorns allmänna hjärna, bra på en sak i taget (serriell beräkning). Används för vanlig programkörning.
    GPU (Graphics Processing Unit) Specialiserad för parallella beräkningar, många små beräkningar i taget. Som sker vid deep learning där tusentals operationer sker samtidigt. Mycket snabbare än CPU vid träning av neurala nätverk.